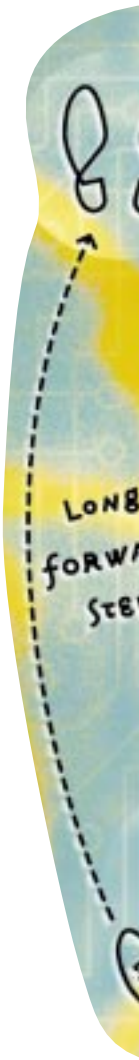
REQUIREM TRACING

MATTHIAS JARKE, *GUEST EDITOR*



In this ever-changing business and technology environment, the risk of inconsistencies in systems development and evolution multiplies. Experience reuse becomes a necessity in order to control quality, costs, and time, even when personnel changes. Requirements tracing is emerging as an effective bridge that aligns system evolution with changing stakeholder needs. It also helps uncover unexpected problems and innovative opportunities, and lays the groundwork for corporate knowledge management.

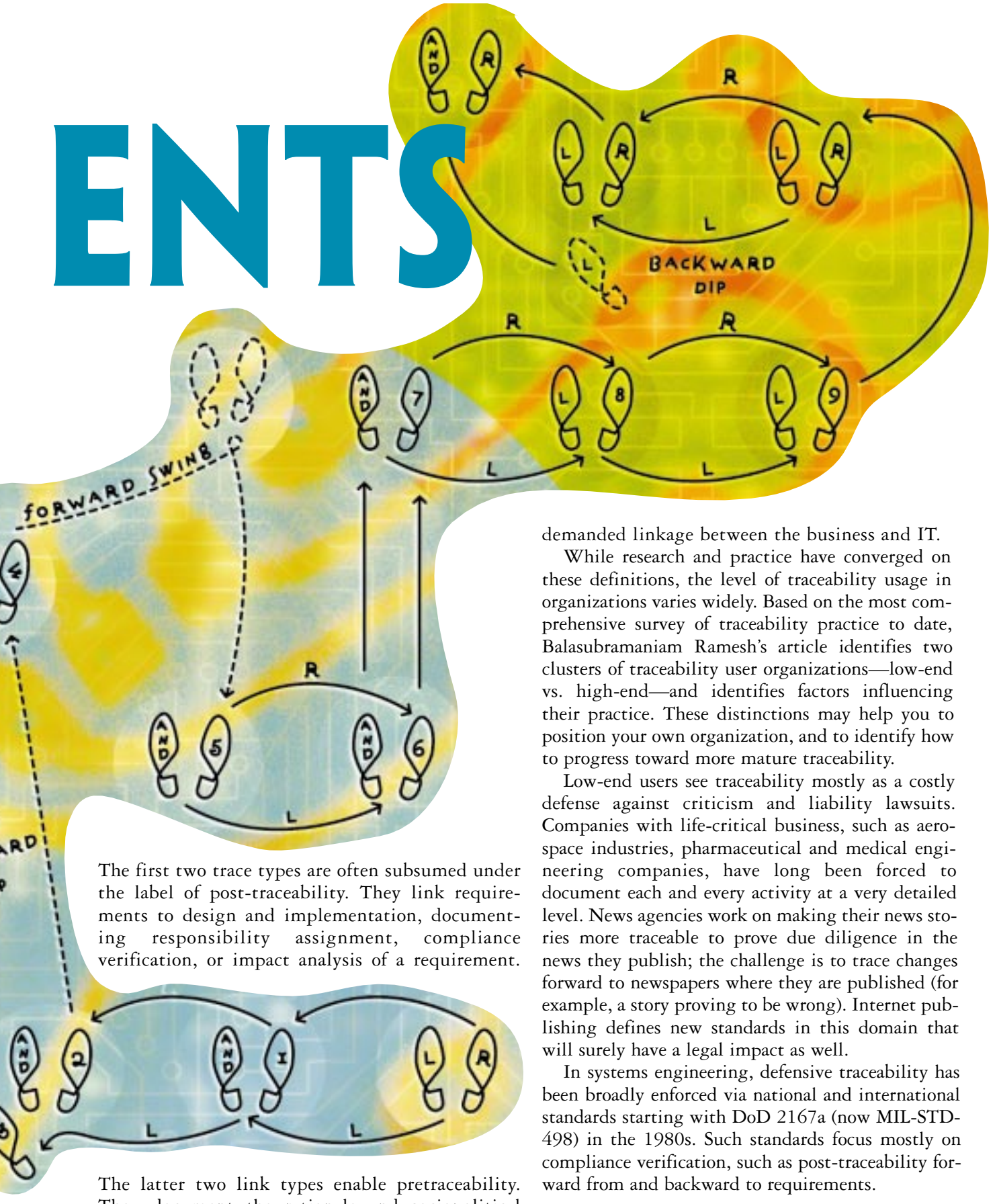
However, few organizations fully recognize—or even understand—the true potential of the new methods and tools in requirements tracing. A network of projects worldwide has investigated the issues, studied advanced industry solutions, and developed research prototypes in order to provide a more coherent view of where we are moving.

Far removed from its time-honored definition as a fuzzy early phase of systems development, requirements engineering is now recognized as the key tool to establish a vision of system-related change in its technical, cognitive, and social context [5]. In other words, it is the task of requirements engineering to proceed along three dimensions: managing the convergence of stakeholder interests toward agreement on key system goals and constraints; achieving a sufficient shared understanding of the issues involved in realizing the system vision, such as its functionality, nonfunctional properties, intended and unintended side effects; and documenting this understanding in adequate representation formats, for human information sharing as well as for computerized system development [6].

The intended result of this process is a structured but evolving set of agreed, well understood, and carefully documented requirements. Requirements traceability, then, is defined as the ability to describe and follow the life of a requirement, in both a forward and backward direction, ideally through the whole systems life cycle [3]. Four kinds of traceability links are typically distinguished with respect to their process relationships to requirements [1]:

- *Forward from requirements.* Responsibility for requirements achievement must be assigned to system components, such that accountability is established and the impact of requirements change can be evaluated.
- *Backward to requirements.* Compliance of the system with requirements must be verified, and *gold-plating* (designs for which no requirements exist) must be avoided.
- *Forward to requirements.* Changes in stakeholder needs, as well as in technical assumptions, may require a radical reassessment of requirements relevance.
- *Backward from requirements.* The contribution structures underlying requirements are crucial in validating requirements, especially in highly political settings.

ENTS



The first two trace types are often subsumed under the label of post-traceability. They link requirements to design and implementation, documenting responsibility assignment, compliance verification, or impact analysis of a requirement.

The latter two link types enable pretraceability. They document the rationale and sociopolitical context from which the requirements emerge. It is fair to say that post-traceability is much better understood than pretraceability, even though only pretraceability will really provide the often demanded linkage between the business and IT.

While research and practice have converged on these definitions, the level of traceability usage in organizations varies widely. Based on the most comprehensive survey of traceability practice to date, Balasubramaniam Ramesh's article identifies two clusters of traceability user organizations—low-end vs. high-end—and identifies factors influencing their practice. These distinctions may help you to position your own organization, and to identify how to progress toward more mature traceability.

Low-end users see traceability mostly as a costly defense against criticism and liability lawsuits. Companies with life-critical business, such as aerospace industries, pharmaceutical and medical engineering companies, have long been forced to document each and every activity at a very detailed level. News agencies work on making their news stories more traceable to prove due diligence in the news they publish; the challenge is to trace changes forward to newspapers where they are published (for example, a story proving to be wrong). Internet publishing defines new standards in this domain that will surely have a legal impact as well.

In systems engineering, defensive traceability has been broadly enforced via national and international standards starting with DoD 2167a (now MIL-STD-498) in the 1980s. Such standards focus mostly on compliance verification, such as post-traceability forward from and backward to requirements.

Traceability as a Corporate Strategy

There is an important relationship of traceability to the Software Engineering Institute's concept of Capability Maturity Model [4]. Repeatability of software

processes, the second level of capability maturity, presupposes understandable traces. Higher maturity levels, such as the so-called "managed software process," also demand the capability to compare traces to plans resulting from defined processes.

High-end users see traceability as an investment in corporate knowledge asset management within and beyond information systems engineering. The value of traces in systems maintenance is highlighted by multimillion-dollar lawsuits between software providers and their customers. A new challenge is the blossoming world of component-oriented systems engineering. As component-based business solutions are installed at customer sites for mission-critical applications, traceability backward to requirements must extend beyond organizational

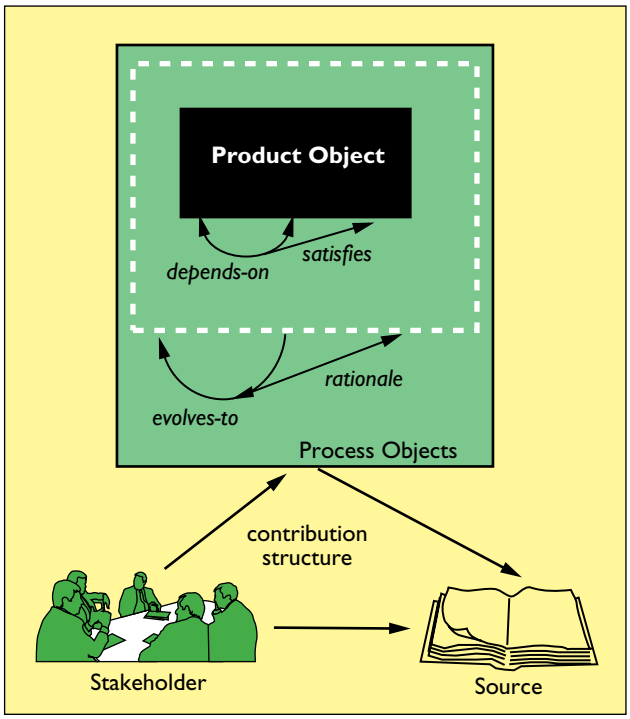


Figure 1. Traceability meta model capturing three dimensions of requirements management

boundaries all the way into the backyard of the component factory, and this over decades! Furthermore, component manufacturers always want to offer best practice components and solutions, and will thus take strong strategic advantage from traceability relationships with their most advanced customers.

Leaders in mechanical engineering, especially in the highly competitive globalized car industry, take ambition and scalability of traceability efforts a step further. As Thomas Rose reports, comprehensive experience assessment is crucial for avoiding costly mistakes in the radical redesign of engineering processes toward time compression, organizational virtualization, and product innovation, for example, when replacing physical car prototypes by computer simulations. Companies are installing process capture and assessment programs across thousands of engineers, in order to accomplish backward traceability to key issues of safety and customer satisfaction. As cars seem to become complex software configurations barely hidden under a gleaming user interface of glass and metal, forward traceability across organizational boundaries into the service garages, or even into the car on the street, becomes another crucial competitive issue.

Traces as a Product

nce it is established what degree of traceability is desirable, it makes sense to define traces as products that satisfy the desired traceability properties. Tracing then is a subprocess of evolutionary system development that supplies and exploits these traces.

Seen as a product, a trace must document all three dimensions of requirements engineering processes mentioned. In the representational dimension, it must capture linkages between the documents produced during a requirements process. Indeed, many early traceability tools either link design documents using a hypertext system, or simply record the links in a spreadsheet independent of the documents themselves.

From a cognitive perspective, a trace captures the conceptual objects, and links them in a meaningful way. For example, a requirement may be linked to the design components responsible for satisfying it by a link, which is annotated by a compliance verification procedure. This kind of traceability dates back to work on dependency-directed reasoning in AI-based design environments in the late 1970s [8], as well in formal approaches to software development in the early 1980s, such as the Knowledge-Based Software Engineering project of the U.S. Air Force.

Finally, a trace should also capture the human cooperation in the design process, that is, how stakeholders contribute to the development and satisfaction of requirements [3]. Many researchers stress the capture of design rationale, that is, decisions made, alternatives considered, underlying assumptions, and stakeholder goals. In yet another kind of process description, objects to be captured

in a trace involve process management information, such as tasks performed, plans followed, and resources consumed.

Many specific link types have been discussed in literature. In order to gain a comprehensive picture, it appears useful to classify all these link types in terms of a common meta model that covers the three dimensions of requirements engineering.

Such a meta model, derived from several comprehensive literature analyses [3, 6] as well as an analysis of traceability schemes used in practice [7], is shown in Figure 1. In this meta model, stakeholders are linked via contribution structures to the conceptual objects they influence, and to the documents in which these objects are recorded.

Concerning the objects, we distinguish those describing products of the design process from those capturing the process itself. Product objects can have the role of requirements or goals, or they can result from trying to satisfy these goals. Each goal object induces constraints that create dependencies between the objects satisfying it; a well-known example is a configuration of software components satisfying some user requirement. The evolution of products, satisfaction and dependency links is described by process objects. They evolve through user actions and design decisions whose rationale is captured in further, auxiliary product objects. An example would be the versioning of a configuration justified by changing technologies and user needs. It is important to note that a trace must contain both product and process objects. This is because the process objects reflect the methodology used, and thus are the starting point for process improvement.

Tracing as a Process

The process of requirements tracing can best be defined in the context of systems development as a whole. Research in workflow management, including software process management, typically distinguishes three domains in which such a process operates: The performance domain in which developers cooperate with stakeholders to do the job; the management domain in which process owners or managers enact process/workflow models to control the work of the performance domain; and the modeling domain in which process engineers define and refine suitable work processes, based on general principles as well as on feedback from both the management and the performance domain [2].

Tracing is the key feedback mechanism between these three domains. Traces provide immediate performance feedback to management, but also longer-term information to the modeling domain where performance traces are analyzed and compared with the plans used by the management domain. Combined with a sufficiently rich semantic structure of the process models themselves, this can establish a computer-supported self-optimizing development process in the organization [4]. However, present development environments support the necessary flexibility in trace capture and usage only in a rudimentary manner, due to various obstacles.

ESTABLISHING AND MAINTAINING REQUIREMENTS TRACEABILITY IS AN EXPENSIVE AND POLITICALLY SENSITIVE ENDEAVOR. DEVELOPERS ARE NOT EXACTLY KNOWN FOR THEIR LOVE OF DOCUMENTATION.

Establishing and maintaining requirements traceability is an expensive and politically sensitive endeavor. Developers are not exactly known for their love of documentation. Traceability should come as a side effect of their daily productive work rather than imposing additional bureaucracy.

Developers may also fear that traces are used against them, for example, for work evaluation or for making specialists superfluous by capturing their best practice in systems. The degree and quality of trace capture, as well as protecting the ownership to traces, are therefore not just technical and

business issues but also political ones. They need to be addressed by a clever combination of policy decisions, incentive schemes, and technical support.

Current traceability tools have made significant inroads into systems engineering practice. However, there are still major gaps. Adaptability to project-specific needs is critical for traceability support environments. Ralf Dömges and Klaus Pohl identify three key requirements that current tools only satisfy to a limited degree: integration into the process to reduce capture effort, adaptation to the situation, and support for organizational knowledge creation. They also present an architecture and a prototype system that satisfies these requirements in a seamless approach.

A Challenge for the Future

The recent emphasis on enhanced traceability is part of a larger vision of *cooperative information systems* emerging from trends in industrial practice as well as academic research. The article by DeMichelis et al. summarizes results from a European-American-Australian-Canadian joint effort that tried to formulate this vision and promote new conference series and government initiatives, including a proposed major U.S. initiative on cooperative computing.

Generalizing the software process architecture discussed here, cooperative information systems are defined as continuously aligning three very different, but interdependent facets: human work practice of the users, organizational concepts and models, and open information systems technologies. This continued alignment is crucial for the agility of an organization—its capability to manage change. Recent advances in conceptual modeling, component-based information technologies, and Internet-enabled group work are making such continued alignment possible. However, current industry solutions tend to overemphasize one of these facets over the others, and researchers are even more locked in traditional community boundaries. The article poses a challenge to research and practice alike: to consider a broad framework of thinking that enables communities such as databases, distributed systems, CSCW, and organizational IS research to take advantage of each others' results and methods.

Traceability—as the means to establish the links between people-system reality and the models under which they are managed—takes center stage in this framework, and the case studies reported this special section provide ample evidence of its potential. C

References

1. Davis, A.M. The analysis and specification of systems and software requirements. In *Systems and Software Requirements Engineering.* IEEE Computer Society Press, 1990, 119–144.
2. Dowson, M. Software process modeling themes and issues. In *Proceedings of the 2nd Int'l. Conference Software Process* (Berlin, Germany, 1993). IEEE Computer Society Press, 54–62.
3. Gotel, O., Finkelstein, A.W. An analysis of the requirements traceability problem. In *Proceedings of the Int'l. Conference Requirements Engineering* (Colorado Springs, Co, 1994), IEEE Computer Society Press, pp. 94–102.
4. Humphrey, W. *Managing the Software Process.* Addison-Wesley, Reading, Mass., 1990.
5. Jarke, M., Pohl, K. Requirements engineering in 2001: (Virtually) managing a changing reality. *IEE Softw. Eng. J. 9*, 6 (1994), 254–263.
6. Pohl, K. *Process Centred Requirements Engineering.* Advanced Software Development Series, Wiley & Sons Ltd., Taunton, England, 1996.
7. Ramesh, B., Jarke, M. Towards reference models for requirements traceability. Technical Report, Georgia State University, submitted for publication.
8. Stallman, R., and Sussman, G. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit design. *Artif. Intell. 9*, 2 (1977), 135–196.

MATTHIAS JARKE (jarke@informatik.rwth-aachen.de) is Professor of Information Systems and Chair of the Computer Science Department at RWTH Aachen, Germany.